

AD-A188 155

FUZZY LOGIC INFERENCE PROCESSOR FOR REAL TIME CONTROL:  
A SECOND GENERATIO.. (U) NORTH CAROLINA UNIV AT CHAPEL  
HILL DEPT OF COMPUTER SCIENCE H WATANABE ET AL. 1986

1/1

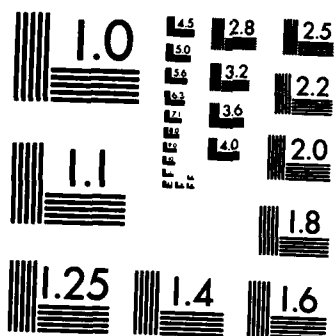
UNCLASSIFIED

N00014-86-K-0680

F/G 25/5

NL





MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A188 155

# Fuzzy Logic Inference Processor for Real Time Control: A Second Generation Full Custom Design

Hiroyuki Watanabe\*

Wayne Dettloff†

Department of Computer Science\*  
University of North Carolina  
Chapel Hill, NC 27599-3175  
Tel. (919) 962-1817

Microelectronics Center of North Carolina†  
Research Triangle Park, NC 27709  
Tel. (919) 248-1874

NOV 23 1987

## abstract

The VLSI implementation of a fuzzy logic inference mechanism allows the use of rule-based control and decision making in demanding real-time applications such as robot control and in the area of command and control. The full custom CMOS VLSI is described. The chip is second generation of the design. It has several design features which make the use of this chip realistic. These features include reconfigurable architecture, on-chip fuzzification and de-fuzzification, and memory and data-path redundancy. The chip consists of 614,000 transistors of which 460,000 are used for RAM memory.

*Keywords: automatic process control systems; chip architecture*

## 1 Introduction

Fuzzy logic based control uses a rule-based expert system paradigm in the area of real-time process control [4]. It has been used successfully in numerous areas including chemical process control, train control [12] cement kiln control [2], and control of small aircraft [5]. In order to use this paradigm of a fuzzy rule-based controller in demanding real-time applications, the VLSI implementation of the inference mechanism has been an active research topic [9,10,11]. Potential applications of such a VLSI inference processor includes real-time decision-making in the area of command and control [3], control of the precision machinery [1], and robotic systems [6].

We have been designing a second-generation VLSI fuzzy logic inference engine on a chip. The new architecture of the inference processor has the following

important improvement compared to previous work:

1. programmable rule set memory
2. on-chip fuzzifying operation - table lookup
3. on-chip defuzzifying operation - center of area algorithm
4. reconfigurable architecture
5. RAM redundancy for higher yield

The original prototype experimental chip (designed at AT&T Bell Labs) had minimal logic on chip. For example, it used ROM for the rule set memory which reduced its utility [10]. We are now designing a more realistic chip which has RAM for the rule set memory so that rules can be programmable. In addition to the fuzzy inference mechanism, the fuzzifying and defuzzifying operations are performed on chip. The new design has a reconfigurable architecture such that we can have either 51 rules, 4 inputs and 2 outputs, or 102 rules, 2 inputs and 1 output. These new design decisions render the new architecture realistic.

## 2 Fuzzy Set and Fuzzy Logic

Fuzzy set is based on a generalization of the concept of the ordinary set. In an ordinary set, we associate a characteristic function for each set. For example, we can define a set  $S$  with its characteristic function  $f_s \rightarrow \{0, 1\}$ . Then, for all  $e$  in the universal set  $U$ ,

$$\begin{aligned} e \in S & \text{ if } f_s(e) = 1, \\ e \notin S & \text{ if } f_s(e) = 0. \end{aligned}$$

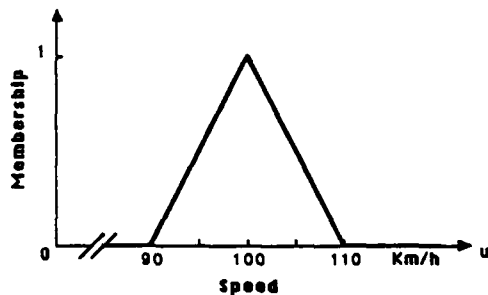


Figure 1: Approximately 100 km/h.

Each element of the universe either belongs to or does not belong to the set  $S$ . In a fuzzy set, an element can be a member of the set with varying degree of membership. The associated characteristic function, therefore, returns any real number between 0 and 1, and it is termed as the membership function. For a fuzzy set  $F$ , we have an associated membership function  $\mu_F(e) \rightarrow [0, 1]$ . For example, if element  $e$  is a member of fuzzy set  $F$  with degree 0.34, the associated membership function returns this value,  $\mu_F(e) = 0.34$ . If  $\mu_F(e) = 0$ ,  $e$  is entirely outside of fuzzy set  $F$ , and if  $\mu_F(e) = 1$ ,  $e$  is entirely inside of fuzzy set  $F$ . Fuzzy set is represented by a set of ordered pairs of an element  $u_i$  and its grade of membership:

$$F = \{(u_i, \mu_F(u_i))\}, u_i \in U$$

where  $U$  is a universe of discourse. Using a fuzzy set, we can represent imprecise and vague concepts and data. For example, *approximately 100 km/h* is represented by the fuzzy set whose membership function is shown in Figure 1. We can extend classical set theory by defining basic set theoretic operations over fuzzy sets. The following definition of intersection and union with fuzzy sets are suggested by Zadeh [13]. The set theoretic operations with fuzzy sets are defined via their membership functions. Let  $A$  and  $B$  be a fuzzy set, then union, intersection and complement of the fuzzy sets are defined as follows. The membership function of the intersection  $C = A \cap B$  is defined by

$$\mu_C(e) = \min(\mu_A(e), \mu_B(e)), e \in U.$$

The membership function of the union  $D = A \cup B$  is defined by

$$\mu_D(e) = \max(\mu_A(e), \mu_B(e)), e \in U.$$

The membership function of the complement  $\neg A$  of  $A$  is defined by

$$\mu_{\neg A}(e) = 1 - \mu_A(e), e \in U.$$

In the traditional logic, one of the most important inference rules is *modus ponens*, that is

|             |             |
|-------------|-------------|
| Premise     | A is true   |
| Implication | If A then B |
| Conclusion  | B is true   |

Here,  $A$  and  $B$  are crisply defined propositions. We can construct a *fuzzy proposition* using a fuzzy set such as:

Current speed is *approximately 100 km/h*.

By introducing fuzzy propositions into *modus ponens*, we can generalize *modus ponens*. Let  $C, C', D, D'$  be fuzzy sets. Then the *generalized modus ponens* states:

|             |                               |
|-------------|-------------------------------|
| Premise     | $x$ is $C'$                   |
| Implication | If $x$ is $C$ then $y$ is $D$ |
| Conclusion  | $y$ is $D'$                   |

We can use different premises to arrive at different conclusions using the same implication. For example,

|             |  |
|-------------|--|
| Premise     | Visibility is <i>slightly low</i>                            |
| Implication | If visibility is <i>low</i><br>then condition is <i>poor</i> |
| Conclusion  | Condition is <i>slightly poor</i>                            |

or

|             |  |
|-------------|--|
| Premise     | Visibility is <i>very low</i>                                |
| Implication | If visibility is <i>low</i><br>then condition is <i>poor</i> |
| Conclusion  | Condition is <i>very poor</i>                                |

The above inference is based on the compositional rule of inference for approximate reasoning proposed by Zadeh [14]. Suppose we have two rules with two fuzzy clauses in the IF-part and one clause in the THEN-part:

- Rule 1: If ( $x$  is  $A_1$ ) and ( $y$  is  $B_1$ ) then ( $z$  is  $C_1$ ),  
Rule 2: If ( $x$  is  $A_2$ ) and ( $y$  is  $B_2$ ) then ( $z$  is  $C_2$ ).

We can combine the inference of the multiple rules by assuming the rules are connected by OR connective, that is Rule 1 OR Rule 2 [10].

Given fuzzy proposition ( $x$  is  $A'$ ) and ( $y$  is  $B'$ ), weights  $\alpha_i^A$  and  $\alpha_i^B$  of clauses of premises are calculated by:

$$\alpha_i^A = \max_x(A', A_i),$$

$$\alpha_i^B = \max_y(B', B_i), \text{ for } i = 1, 2.$$

Then, weights  $w_1$  and  $w_2$  of the premises are calculated by:

$$w_1 = \min(\alpha_1^A, \alpha_1^B),$$

$$w_2 = \min(\alpha_2^A, \alpha_2^B),$$

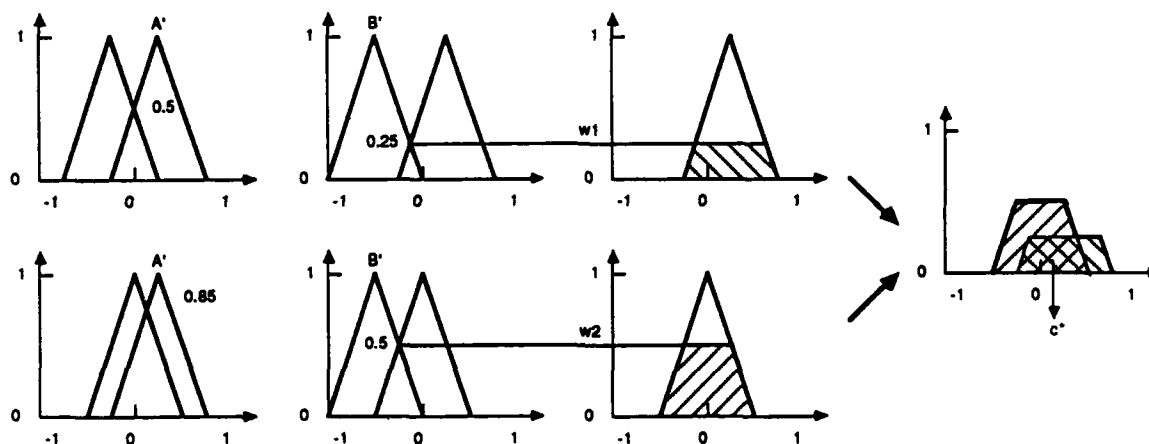


Figure 2: Inference.

Weight  $\alpha_i^A$  represents the closeness of proposition ( $x$  is  $A_i$ ) and proposition ( $x$  is  $A'$ ). Weight  $w_i$  represents similar measure for the entire premise for the  $i^{th}$  rule. The conclusion of the first rule is

$$C'_1 = \min(w_1, C_1),$$

The conclusion of the second rule is

$$C'_2 = \min(w_2, C_2),$$

The overall conclusion  $C'$  is obtained by

$$C' = \max(C'_1, C'_2).$$

This inference process is shown in Figure 2. In this example,  $\alpha_1^A = 0.5$  and  $\alpha_1^B = 0.25$ , therefore  $w_1 = 0.25$ .  $\alpha_2^A = 0.85$  and  $\alpha_2^B = 0.5$ , therefore  $w_2 = 0.5$ .

### 3 Rule-based controller

The usual approach for automatic process control is to establish a mathematical model of the process. However, this is not always feasible. In some cases, there is no proper mathematical model because the process is too complex or ill-understood. In other cases, experimenting with plants for construction of mathematical models is too expensive. In still other cases, the mathematical models are too complicated or computationally expensive and are not suitable for real time use. For such processes, however, skilled human controllers may be able to operate the plant satisfactorily. The operators are quite often able to express their operating practice in the form of rules which may be used in a rule-based controller. The rule based controllers model the behavior of the expert human operator instead of the process. The following is a rule from an aircraft flight controller [5].

This rule takes three inputs and has two outputs.

- If
- (1) The rate of descent is *Positively Medium*,
  - (2) The airspeed is *Negatively Big* (compared to the desired airspeed),
  - (3) The glide slope is *Positively Big* (compared to the desired slope).

Then (1) change engine speed by *Positively Big*, and

- (2) change elevator angle by *Insignificant Change*.

The expressions, *Positively Medium*, *Positively Big*, *Insignificant Change*, and others represent imprecise amounts. They represent intuitive feel of the expert human controller. They correspond to the imprecise expressions used by the expert for communicating a rule of thumb. They are represented by using fuzzy sets and their associated membership functions.

The fuzzy set, such as *Positively Medium* is represented by the membership function over an appropriate universe of discourse such as revolutions per minute (rpm). The possible definitions of fuzzy sets are shown in Figure 3. The control rules are encoded using typically 10 to 70 rules. The Control is performed based on the fuzzy inference mechanism described in Section 2 and Figure 2. In controlling a process, all of the rules are compared to the current inputs (observations) and fired. The actions (THEN-part) of each rules are weighted by how close its IF-part matches the current observation. In the example of Figure 3, a rule has two inputs and a single output. The weights are represented by  $w_1$  and  $w_2$ . The results of firing of each rule are then combined by superimposing them. The final result which is supplied to a controller should be a crisp

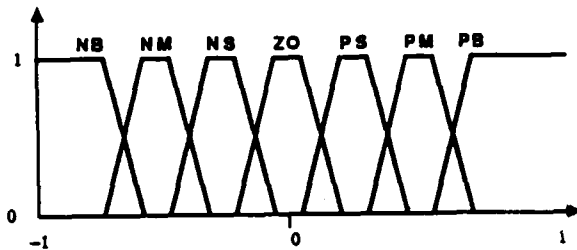


Figure 3: Typical fuzzy sets.

number rather than a fuzzy set, therefore we need to perform a defussifying operation. This is computed by taking a center of area under the fuzzy membership function of the final result. Even though each individual rule is an incomplete rule of thumb, the results of firing each rule are properly weighted and combined and the final result represents reasonable compromise.

In order for VLSI implementation of fuzzy inference to be useful, a fair amount of pre-processing (fuzzifying) and post-processing (defussifying) must be performed on chip. The AT&T prototype chip assumed that both of these processes are performed by the host-processor. However, the inference processing is too fast for fuzzifying and defussifying to take place off-chip by a host processor. This assumption burdened the host processor and nullified the advantage of VLSI implementation of the inference mechanism.

#### 4 Chip Architecture and Implementation

The process controller system is configured as in Figure 4. The VLSI implementation is done with four components; a fuzzifier, a rule memory, an inference mechanism, and a defussifier on a single chip. Each input and output data item is 6 bits. This fits well with available A/D and D/A converters. In addition, our chip will communicate with a host processor. The chip has three stage pipelining architecture. The pipeline consists of IF-part, THEN-part, and defussifier.

We considered the size of the fuzzy set and the grade of fuzziness for practical use. In most cases, a fuzzy variable has three to sixteen elements and the grade of fuzziness has three to twelve levels [5,8]. In this chip implementation, the universe of discourse of a fuzzy set is a finite set with 64 elements (i.e. 6 bits). The membership function has 16 levels (i.e. 4 bits). That is, 0 represents no membership, 15 represent full membership, and other numbers represent points in the unit interval [0, 1]. A fuzzy membership function is, therefore, discretised using 64 numbers of 4 bit; that is 256 bits of memory storage. The representation of a fuzzy set is as follows:

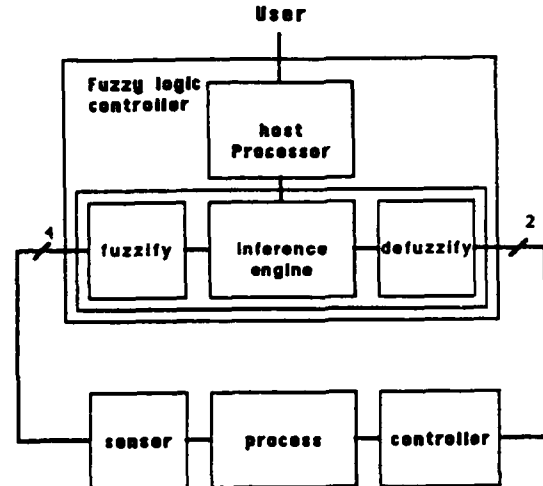


Figure 4: Fuzzy logic controller.

|       |       |     |              |     |          |
|-------|-------|-----|--------------|-----|----------|
| $u_0$ | $u_1$ |     | $u_i$        |     | $u_{63}$ |
| 0000  | 0011  | ... | $\mu_F(u_i)$ | ... | 0000     |

Fuzzifying is done using a table look-up. For each observation (i.e. input stream), we store a table of the membership function normalised at the center of the horizontal axis. That is, the full membership is at the center. According to an input value, the membership function is shifted. The chip can produce 64 different membership functions from a single stored pattern. The membership function can be associated with a predicted measurement error of a sensor. If we do not need fuzziness in the observed value, we can store a pulse function, that is only one entry has membership 1 and all the other entries have 0's. The result of the fuzzifying is broadcasted to all of the rules. In the actual chip implementation, the content of the table is not shifted. Rather a starting address for table look-up is shifted according to an observation input.

The chip is re-configurable. A control system can take four inputs and produce two outputs or take two inputs and produce one output according to an application. With the first configuration, we can have 51 rules on a single chip. Each rule has four clauses in the IF-part and two actions in the THEN-part.

If A and B and C and D  
Then Do E, and  
Do F.

With the second configuration, we can execute 102 rules using a same data-path. Each rule has two clauses in the IF-part and one action in the THEN-part.

If A and B Then Do E,  
If C and D Then Do F.

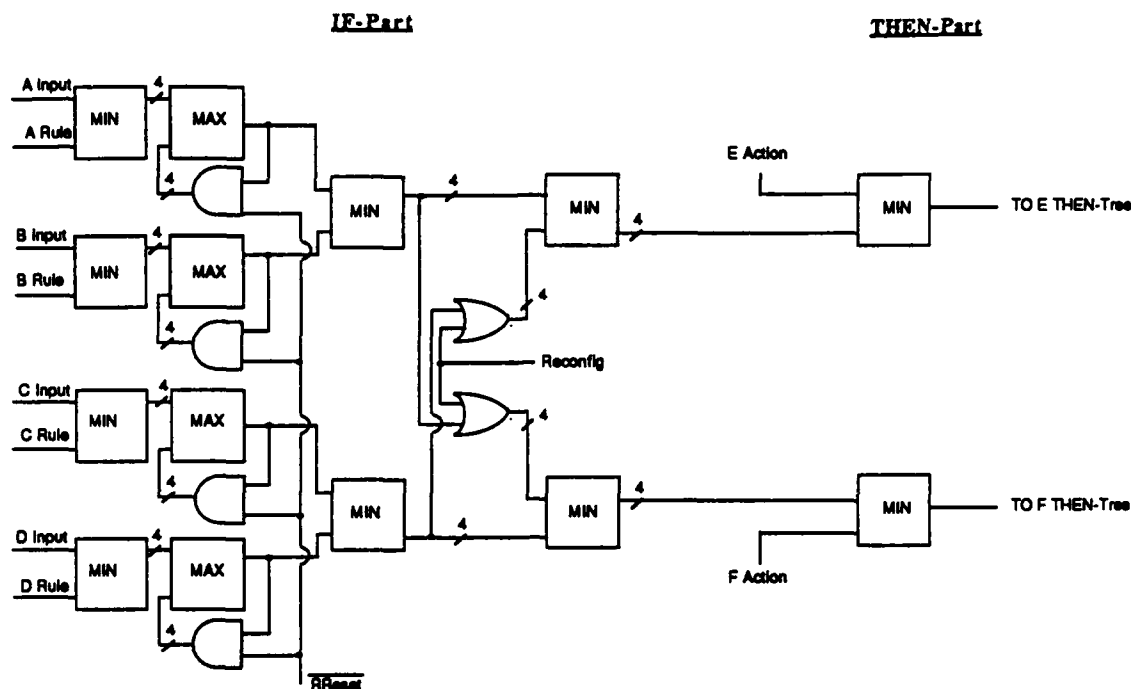


Figure 5: Reconfigurable data-path for rule execution.

A data-path is assigned for each rule, therefore all of 51 or 102 rules are executed in parallel. There are only two basic units; they are a parallel minimum unit and a parallel serial unit. The former performs the intersection operation on fuzzy sets, and the latter performs the union operation. The configuration of the If-part of the data-path is shown in figure 5. The data-path can execute one rule with 4 if-clauses or two rules with 2 if-clauses. Four pairs of min/max units compute the weight  $\alpha$ 's for each clause. The min elements organised as a binary tree compute weights  $w$  of the premise which is the minimum of all  $\alpha$ 's. In the 51 rule configuration, the last two minimum units compute the same weight  $w_i$ . In the 102 rule configuration, streams of 1's are supplied and these two min elements behave as delay elements. The control of configuration is done by setting a bit in the status register from the host computer. Defussifying is done by computing a center of area (COA) under the final membership function. Denoting the final fuzzy subset as A, the COA algorithm computes the following:

$$c^* = \frac{\sum_{n=0}^{63} n \cdot \mu_A(n)}{\sum_{n=0}^{63} \mu_A(n)}$$

Since each element of the universe is processed serially, we can substitute multiple addition for multiplication in the above computation. The data sequence

from the THEN-part is produced starting from the most significant data point as follows:

$$\mu_A(63), \mu_A(62), \dots, \mu_A(1), \mu_A(0).$$

Two adders and two registers are used as shown in Figure 6. The numerator is computed by the first adder and denominator is produced by the second adder. The denominator is computed as by repeated addition of the result of the first adder by the second adder which computes the following formula.

$$\begin{aligned} \sum_{n=0}^{63} n \mu_A(n) = & \mu_A(63) + \\ & \mu_A(63) + \mu_A(62) + \\ & \mu_A(63) + \mu_A(62) + \mu_A(61) + \\ & \vdots \\ & \mu_A(63) + \mu_A(62) + \mu_A(61) + \dots + \mu_A(0). \end{aligned}$$

In order to achieve higher yield, we allocated 51 data-paths on the chip, and non-functioning memory units and data-paths can be isolated from the rest of the chip. The isolation is achieved by blowing a fuse using laser technology. Each pair of a memory unit and a data-path can be reprogrammed to any other

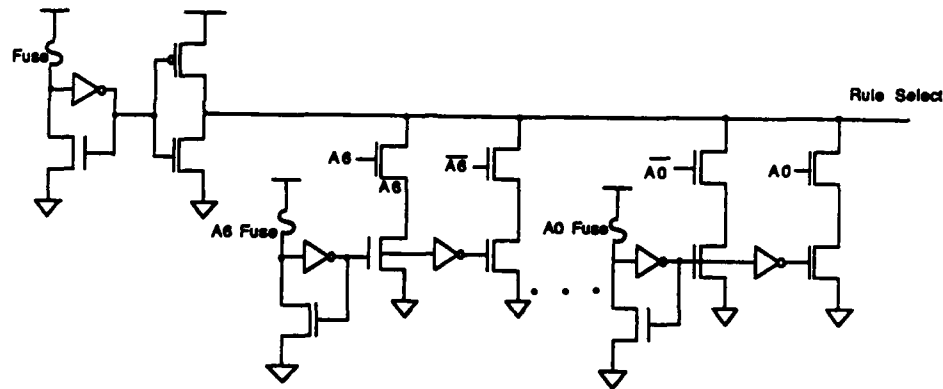


Figure 7: Redundancy

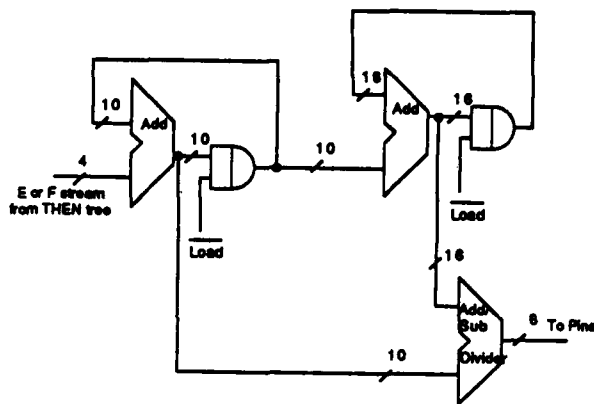


Figure 6: Defussifier circuit.

address also by blowing a fuse. This allows a continuous addressing of memory/data-paths after removal of a defective unit from a chip. The schematic diagram for address removal and re-programming circuit is shown in Figure 7.

The host processor down loads the rule set and table for fuzzification at start up time. The fuzzy processor looks like a static RAM chip to the host processor. The RAM system, however, only has a row decoder and does not have a column decoder. A user can address each row (corresponds a clause/action of a rule) by a memory address register. Each column is addressed by a shift register because data are accessed sequentially. The last address is reserved and mapped to the status register. This register controls the configuration of data-paths and operational modes (load, run, or test).

The chip is designed for a 1  $\mu\text{m}$  N-well CMOS process of MCNC [7]. It uses non-overlapping two phase clocking scheme. The chip is designed with a target operational speed of 40MHz. The chip consists from approximately 614,000 transistors of which about 470,000 are used to form the static RAM system. The die size is 7750 $\mu\text{m}$  by 9080 $\mu\text{m}$ , and is packaged in a standard pin grid array with 84 pins. The supply voltage is 3.0–3.3 v.

## 5 Acknowledgement

The defussifying circuits were designed by Jeff Hultquist and Jih-Fang Wang of the University of North Carolina at Chapel Hill. Kathy E. Yount of MCNC assisted for designing VLSI layout and producing figures for publication. Research reported here is supported in part by Microelectronic Center of North Carolina through MCNC Design Initiative Program and by the Office of Naval Research (Contract No. N00014-86-0680).



## References

- [1] Burg, B., L. Foulloy, J. C. Heudin, and B. Zavidovique, "Behavior Rule Systems for Distributed Process Control," *Proc. of 2nd Conf. on AI Applications*, pp. 189-203, December 1985.
- [2] Holmblad, L. P. and J. J. Ostergaard, "Control of a Cement Kiln by Fuzzy Logic," in *Fuzzy Information and Decision Processes*, M. M. Gupta and E. Sanchez (Eds), pp. 389-399, 1982.
- [3] Kawano, K., M. Kosaka, and S. Miyamoto, "An Algorithm Selection Method Using Fussy Decision-Making Approach," *Trans. Society of Instrument and Control Engineers*, Vol. 20, No. 12, pp. 42-49, 1984. (in Japanese)
- [4] King, P. J. and E. H. Mandani, "The Application of Fussy Control Systems to Industrial Processes," *Automatica*, Vol. 13, No. 3, pp. 235-242, 1977.
- [5] Larkin, L. I., "A Fussy Logic Controller For Aircraft Flight Control," in *Industrial Applications Of Fuzzy Control*, M. Sugeno (Ed), pp. 87-103, 1985.
- [6] Murakami, S., F. Takemoto, H. Fujimura, and E. Ide, "Weld-line Tracking Control of Arc Welding Robot Using Fussy Logic Controller," *Proc. of 2nd Inter. Fuzzy Systems Association Congress*, pp. 353-357, July 1987.
- [7] Sharma, D., S. Goodwin-Johansson, D. S. Wen, C. K. Kim, and C. M. Osburn, "A  $1\mu\text{m}$  CMOS Technology with Low Temperature Processing," *Extended Abstracts of 171 meetings of the Electrochemical Society*, Vol. 87-1, pp. 213-214, May 1987.
- [8] Sugeno, M. and Murakami, K., "Fuzzy Parking Control of Model Car," *Proc. the 23rd IEEE Conf. Decision and Control*, December 1984.
- [9] Togai, M, and S. Chiu, "A Fuzzy Logic Accelerator and a Programming Environment for Real-Time Fuzzy Control," *Proc. of 2nd Inter. Fuzzy Systems Association Congress*, pp. 147-151, July 1987.
- [10] Togai, M. and H. Watanabe, "An Inference Engine for Real-time Approximate Reasoning: Toward an Expert on a Chip," *IEEE EXPERT*, Vol. 1, No. 3, pp. 55-62, August 1986.
- [11] Yamakawa, T. and T. Miki, "The Current Mode Fuzzy Logic Integrated Circuits Fabricated by the Standard CMOS Process," *IEEE Transactions on Computers*, Vol. C-35, No. 2, pp. 161-167, February 1986.
- [12] Yasunobu, S., S. Miyamoto, T. Takaoka, and H. Ohshima, "Application of Predictive Fuzzy Control to Automatic Train Operation Controller," *Proc. of IECON'84*, pp. 657-662, 1984.
- [13] Zadeh, L. A., "Fuzzy set," *Information and Control*, Vol. 8 pp. 338-353, 1965.
- [14] Zadeh, L. A., "Outline of a New Approach to the Analysis of Complex Systems and Decision-Making Approach," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SME-3, pp. No. 1, pp. 28-45, January 1973.

|                     |                                     |
|---------------------|-------------------------------------|
| Accession For       |                                     |
| NTIS GRA&I          | <input checked="" type="checkbox"/> |
| D110 TAB            | <input type="checkbox"/>            |
| Unannounced         | <input type="checkbox"/>            |
| Justification       | <input type="checkbox"/>            |
| By _____            |                                     |
| Distribution/ _____ |                                     |
| Availability Codes  |                                     |
| Dist                | Avail and/or Special                |
| AI                  |                                     |



END

FILMED

FEB. 1988

DTIC